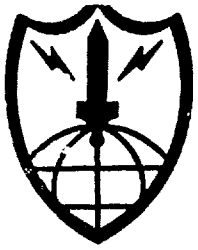

USAISEC

US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES

Software Development Information Supported by the SEI Contractor Assessment Questionnaire

March 1991

ASQB-GI-91-015

AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION / AVAILABILITY OF REPORT N/A
2b. DECLASSIFICATION / DOUWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ASQB-GI-91-015	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION Purdue University / SERC 6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION N/A
6c. ADDRESS (City, State, and ZIP Code) Department of Computer Science West LaFayette, Indiana 47907	7b. ADDRESS (City, State, and Zip Code) N/A
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AIRMICS 8b. OFFICE SYMBOL (if applicable) ASQB - GI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 62783A PROJECT NO.: DY10 TASK NO.: 02-04-02 WORK UNIT ACCESSION NO.:

11. TITLE (Include Security Classification)
Software Development Information Supported by the SEI Contractor Assessment Questionnaire (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Dunsmore, Buster; Varnau, Steve

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
final report	FROM _____ TO _____	1991, March, 12	25

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	CASE Tools; Distributed Computing Design System; DCDS; Teamwork; Excelerator; EPOS; DesignAid; SA Tools; Software engineering environment Systems; SEES; Ada Programming Support Environment; APSE

19. ABSTRACT (Continue on reverse if necessary and identity by block number)

The information presented in this report was acquired as part of the Distributed Computing Design System (DCDS) evaluation project, AIRMICS Report ASQB-GI-91-009, "Evaluation of DCDS for Meeting the Data Collection Requirements for Software Specification, Development, and Support". The DCDS evaluation is outlined in the succeeding paragraph. This report augments the DCDS report by comparing the data requirements for a fully flexible CASE environment with the data requirements suggested by the Software Engineering Institute (SEI) Contractor Assessment Questionnaire.

The DCDS evaluation technical report consists of five separate but related reports which evaluate the Distributed Computing Design System (DCDS). DCDS was developed by TRW as a software development environment for real-time, distributed systems. The principal investigator evaluated DCDS in terms of: a) its data collection requirements, b) its software development information completeness, c) its usability, d) how it compares to five commercially available CASE tools, and e) its suitability as an Ada Programming Support Environment (APSE).

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED / UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Howard C "Butch" Higley	(404) 894-3110	ASQB-GI

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

The research herein was performed for the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), the RDTE organization of the U.S. Army Information Systems Engineering Command (USAISEC). The sponsor for the project was the Office of the Director of Information Systems for Command, Control, Communications, and Computers (ODISC4). The principal investigator was Dr. H. Dunsmore of Purdue University.

This research report is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited, and is not protected by copyright laws. Your comments on all aspects of the document are solicited.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

s/ Glenn E. Racine
Glenn E. Racine
Chief
CISD

s/ John R. Mitchell
John R. Mitchell
Director
AIRMICS

Software Development Information Supported by the SEI Contractor Assessment Questionnaire

S. Varnau
H. Dunsmore

Software Engineering Research Center (SERC)
Department of Computer Sciences
Purdue University, West Lafayette, IN 47907

SERC-TR-78-P
July, 1990

Technical Report 3.2 from the Research Project:
Evaluation of DCDS for Meeting the Data Collection
Requirements for Software Specification, Development, and Support

Abstract

In this paper we examine the Software Engineering Institute Technical Report CMU/SEI-87-TR-23 (Fall, 1987). Written by Watts Humphrey and William Sweet, "A Method for Assessing the Software Engineering Capability of Contractors" details questions and procedures to be used in considering potential DoD contractors. Humphrey and Sweet designed a questionnaire to assess objectively the capability of contractors to use modern software engineering techniques in product development. In our previous work we identified data collection requirements for CASE systems. In this paper we compare our data requirements to the data requirements suggested by the SEI questionnaire.

The SEI questionnaire focuses on the software development process including CASE systems as only a part of software engineering capability. It emphasizes two dimensions of software engineering to assess contractor capability - **process maturity** and **technology**. It contains no direct correlation to the **Product** data category that figures prominently in our information requirements. Thus, it is not surprising that the SEI questionnaire scored relatively low in the Product Description and Product Implementation categories. It is also weak in the Product Maintenance categories.

Because of its concentration on the software development process, the SEI report scored relatively high in the Process Management, Process Coordination, and some parts of the Process Quality Control categories. Also, since verification (i.e.,

testing) is very important to the software development process, the SEI questionnaire is fairly strong in the Product Verification categories.

Background

The purpose of this research is to evaluate the Distributed Computing Design System (DCDS) to determine how appropriate it is for a software engineering support environment.

There is increasing interest in software engineering support environments (frequently known as Computer Aided Software Engineering (CASE) systems). There are hundreds of such systems on the market (and more becoming available each year). But, many potential users question how much these support environments increase productivity, if any. Some companies have attempted to use software engineering support environments with mixed success. The overhead costs of purchase, installation, and training coupled with the possibility of short-term decrease in productivity represent serious concerns.

It is clear that no one support environment contains all possible functions, methods, and tools that could be used in software development. Each support environment supports a subset - some far richer than others.

It is becoming increasingly clear that in order to achieve the software productivity gains necessary for world-wide competitiveness that something on the order of software engineering support environments must be used. Thus, research concerning problems with existing software engineering environments and demonstrating alterations that will increase their usefulness will be a valuable contribution to software productivity.

In September, 1987, the Software Engineering Institute published a Technical Report entitled "A Method for Assessing the Software Engineering Capability of Contractors" [HUMP87]. This SEI questionnaire details questions and procedures to be used in considering potential DoD contractors. They designed a questionnaire to assess objectively the capability of contractors to use modern software engineering techniques in product development.

In our previous work, we identified data collection requirements for CASE systems [VARN90]. We now compare these data requirements to the data requirements suggested by the SEI questionnaire. The purpose of this phase of our study is to provide further clarification, extension, and validation of our data requirements from another important source in the software engineering field.

The data collection requirements consist of product data (which describe the software product itself) and process data (which reflect the activity involved in developing and supporting the product). The main focus of these categories and their component sub-categories is the information associated with a particular software project. The SEI questionnaire, on the other hand, evaluates levels of technology and process maturity. The focus here is the process across software projects. This viewpoint includes CASE systems as only a part of software engineering capability. The data requirements viewpoint centers around a flexible CASE environment which supports as much of the software engineering capability as possible. These two views definitely have a large intersection, but the questionnaire is certainly a more abstract perspective.

Results

As stated above, the SEI questionnaire emphasizes two dimensions of software engineering to assess contractor capability. The stages of **process maturity** are used as a major qualifier of contractors. This corresponds to the **Process** category of our data requirements. Process maturity, however, stresses using process data from project to project to improve software development procedures. This concept specifically corresponds to **Project History** data under our **Process Quality Control** category and **Process Plan** data under our **Process Management** category. The stages of **technology** constitute the other major qualifier. This corresponds somewhat to the extent to which a CASE system exhibits an open architecture - which we discussed in our previous report [VARN90].

The SEI questionnaire contains no direct correlation to the **Product** data category that figures prominently in our information requirements. Certainly the questionnaire deals with this type of information indirectly, mainly in the way a development process deals with it. This type of information is critical to software development. However, the SEI questionnaire takes much of this information for granted and deals at a higher level of abstraction. The types of data used in traditional CASE tools are largely ignored.

Below we present a comparison of the relative use of the data requirements identified (see the Appendix). The Henderson/Cooprider column represents data requirements identified from a report describing a functional view of CASE technology [HEND88]. The CASE tool column represents a "best case" combination of all the CASE tools studied in a previous phase of this research [VARN90]. For each category, the given score is the maximum score of all five tools in the previous report. The best cases of those popular tools reflect current technology. The SEI questionnaire column represents data requirements needed to support modern software engineering techniques as described in the SEI questionnaire [HUMP87].

The following scale is used to indicate how completely each requirement is met, or is used by each source.

- == No support at all or not addressed by tool (equivalent to 0)
1 == Possible to incorporate information, but not specifically supported
2 == Category addressed, but not fully supported
3 == Adequate
4 == Exceptional treatment of category
5 == Could not be better

Note that just because two items receive the same number for the same category, this does not mean that they are functionally equivalent for that category. For example, both the CASE Tools and the SEI Questionnaire rate a 3 for the category Description/Implementation below. This does not mean that the CASE Tools and the SEI Questionnaire have identical methods by which the developer can describe the relationships among planned and implemented components - only that we consider the Description/Implementation category as "Adequate" for both the CASE Tools and the SEI Questionnaire.

Also, note that the SEI Questionnaire was intended for a different purpose than that for which we are using it. The reader should not interpret our comments on missing or inadequate information in the SEI Questionnaire as criticisms of this work, which was quite good. Our comments only refer to the adequacy of the SEI Questionnaire for our purposes.

PRODUCT - Description

Category	Henderson/Cooprider	CASE Tools	SEI Questionnaire
Functionality	3	4	2
Interfaces	2	4	2
Performance	3	4	3
Time Constraints	1	3	1
Fault Tolerances	1	2	1
Data Flow	3	4	1
Process Flow	3	4	1
Resources	1	3	1
Structure	4	4	1
Entity-Relation.	2	3	1
Communication	3	4	2
Data	3	3	1
Req./Design	3	3	3
Design/Perf.	3	2	2
Descrip./Impl.	4	3	3
Design/Design	1	3	2
Prototypes	2	2	3
Mean Scores	2.5	3.2	1.8
Range	1-4	2-4	1-3
Inadeq. Pctage.	41%	18%	76%

Note that the *Mean Scores* average the ratings for each item (Henderson/Cooprider report, CASE Tools, and SEI Questionnaire) for all categories. The ***Range*** gives an idea of the variability of that item across all categories. The ***Inadeq. Pctage.*** line reports the percentage of all ratings for each item that are 0, 1, or 2 (instead of 3 or 4 - there were no 5's) and thus judged to be *inadequate*.

To summarize, supporting this category of information we found:

The Henderson/Cooprider (MIT) Report had a 2.5 average score indicating nearly-adequate support for this category. The "best case" from the five CASE tools had a 3.2 average score indicating an adequacy not surprising in light of the intended purpose of these tools, but [VARN90] shows that no one individual tool has a mean greater than 2.6. The SEI Questionnaire had only a 1.8 average score. The SEI questionnaire scored relatively low in this area. This was expected because the focus of the SEI questionnaire is the process - not the product.

PRODUCT - Implementation

Category	Henderson/Cooprider	CASE Tools	SEI Questionnaire
Actual Product	3	3	1
Metrics	2	1	3
Library	4	2	2
Templates	3	3	1
Compile Param.	-	-	-
Mean Scores	2.4	1.8	1.4
Range	0-4	0-3	0-3
Inadeq. Pctage.	40%	60%	80%

To summarize, supporting this category of information we found:

The Henderson/Cooprider (MIT) Report had a 2.4 average score indicating nearly-adequate support for this category. The "best case" from the five CASE tools had a 1.8 average score indicating very little support for this category. The SEI Questionnaire had only a 1.4 average score. The SEI questionnaire is less than adequate for all categories except Metrics. Once again, note that there is no *information collection support for the Compile Parameters category.* This category constitutes information that we believe should be part of a software development environment that appears in none of the Henderson and Cooprider (MIT) Report, the representative CASE tools, or the SEI Questionnaire.

PRODUCT - Verification

Category	Henderson/Cooprider	CASE Tools	SEI Questionnaire
Test Plan	-	3	3
Test Tools	-	-	2
Test Suites	-	-	2
Status	-	1	3
Errors Found	1	2	3
Ver./Descrip.	-	2	2
Analysis	1	2	4
Mean Scores	0.3	1.4	2.7
Range	0-1	0-3	2-4
Inadeq. Pctage.	100%	86%	43%

To summarize, supporting this category of information we found:

The Henderson/Cooprider (MIT) Report had a 0.3 average score indicating almost no support for this category. The "best case" from the five CASE tools had only a 1.4 average score indicating a sad lack of support for verification capabilities in existing tools. On the other hand, the SEI Questionnaire had a 2.7 average score. Verification data is very important to the software development process of a particular project. The SEI questionnaire is fairly strong in this area, especially stressing analysis of project data.

PRODUCT - Maintenance

Category	Henderson/Cooprider	CASE Tools	SEI Questionnaire
Maintenance History	2	2	2
Special Cases	-	1	-
Complaints	-	3	1
Proposed Changes	-	3	1
General Information	-	-	-
Mean Scores	0.4	1.8	0.8
Range	0-2	0-3	0-2
Inadeq. Pctage.	100%	60%	100%

To summarize, supporting this category of information we found:

The Henderson/Cooprider (MIT) Report had a paltry 0.4 average score indicating almost complete lack of support for the maintenance category. The "best case" from the five CASE tools had a not-much-better 1.8 average score indicating poor support for maintenance activities in existing tools. The SEI Questionnaire had a 0.8 average score. The SEI questionnaire concentrates on software development, and like other sources, it is weak in this area.

PROCESS - Management

Category	Henderson/Cooprider	CASE Tools	SEI Questionnaire
Schedule	4	4	3
Budget	-	3	3
Pers. Assign.	4	3	2
Environ. Custom.	3	2	1
Format Parameters	3	4	-
Process Plan	-	-	4
Mean Scores	2.3	2.7	2.2
Range	0-4	0-4	0-4
Inadeq. Pctage.	33%	33%	50%

To summarize, supporting this category of information we found:

The Henderson/Cooprider (MIT) Report had a 2.3 average score indicating some, but very little, support for the process management category. The "best case" from the five CASE tools had a 2.7 average score indicating nearly adequate support for this category. The SEI Questionnaire had a 2.2 average score. Some parts of this area are covered quite well by the SEI questionnaire (e.g., Process Plan), but some project specific data is missing in the personnel category. Two of the categories are tool details, and are lacking. Process planning and improvement is a major concern of the SEI questionnaire.

PROCESS - Coordination

Category	Henderson/Cooprider	CASE Tools	SEI Questionnaire
Project Direct.	3	3	1
Configuration	3	3	3
Standards	3	-	4
Communication	3	1	1
Commun. Formats	3	-	-
Mean Scores	3.0	1.4	1.8
Range	3	0-3	0-4
Inadeq. Pctage.	0%	60%	60%

To summarize, supporting this category of information we found:

The Henderson/Cooprider (MIT) Report had a 3.0 average score (all 3's) indicating uniformly adequate support for the process coordination category. The "best case" from the five CASE tools had only a 1.4 average score indicating a sad lack of support for coordination capabilities in existing tools. The SEI Questionnaire had a 1.8 average score. The SEI questionnaire is very strong in standards and configuration categories. It is very weak in the other areas which may not be considered formal parts of the process, but are still important to quality and productivity.

PROCESS - Quality Control

Category	Henderson/Cooprider	CASE Tools	SEI Questionnaire
Quality Goals	3	-	1
Fault Conseq.	-	2	1
Target Environ.	2	-	1
Inspections	-	-	4
User Input	2	1	-
References	3	2	-
Project History	1	1	4
Mean Scores	1.6	0.9	1.6
Range	0-3	0-2	0-4
Inadeq. Pctage.	71%	100%	71%

To summarize, supporting this category of information we found:

The Henderson/Cooprider (MIT) Report had a 1.6 average score indicating poor support for this category. The "best case" from the five CASE tools had only a 0.9 average score indicating a pitiful lack of support for quality control capabilities in existing tools. The SEI Questionnaire had only a 1.6 average score. The questionnaire relies on inspections, project history, standards, and process maturity to provide quality control. It is quite strong in those areas, but weak in others.

SUMMARY

The table below summarizes the mean scores from the previous 7 tables:

MIT	CASE	SEI	Category
2.5	3.2	1.8	Product Description
2.4	1.8	1.4	Product Implementation
0.3	1.4	2.7	Product Verification
0.4	1.8	0.8	Product Maintenance
2.3	2.7	2.2	Process Management
3.0	1.4	1.8	Process Coordination
1.6	0.9	1.6	Process Quality Control
1.9	2.2	1.8	Mean Scores

From these varied sources we believe we have compiled a very comprehensive data requirements list for CASE tools. We think we have also developed a good evaluation basis for software environments and information repository standards flexible enough to support the fast-paced changes of software engineering technology.

So far in our work we have determined that the software development information as outlined by Henderson and Cooprider in their MIT Report, the actual information supported by existing CASE tools, and the software development information suggested by Humphrey and Sweet in their SEI questionnaire all could do a **much** better job of supporting software development. Our research to this point suggests that the current state-of-the-art in CASE technology is not adequate to provide the kind of software development support needed to meet current data collection requirements for software specification, development, and support.

In the next step of our work we will proceed to compare and contrast the information collected by DCDS with these other sources.

References

[HEND88] Henderson, John C. and Jay G. Cooprider. "Dimensions of I/S Planning and Design Technology". Center for Information Systems Research Technical Report. MIT Sloan School of Management. September, 1988.

[HUMP87] Humphrey, Watts S. and William L. Sweet. "A Method for Assessing the Software Engineering Capability of Contractors". Software Engineering Institute Technical Report CMU/SEI-87-TR-23. Fall, 1987.

[VARN90] Varnau, S. and H. Dunsmore. "Software Development Information Supported by Typical CASE Tools". Software Engineering Research Center Technical Report TR-77-P. July, 1990.

APPENDIX - Data Collection Requirements for Software Specification, Development, and Support

Below we re-present the data requirements identified in our previous report [VARN90]. These requirements are divided into two categories. **Product** data includes everything which describes the software product itself. The typical results of a software project are the requirements, specifications, design, implementation, code metrics, test plans, etc. These materials comprise product data. **Process** data includes everything which reflects the activity involved in developing and supporting the product. This includes personnel, schedule, budget, etc.

Product data is further subdivided into **description**, **implementation**, **verification**, and **maintenance** categories. Description data consists of information from the development phases commonly known as requirements, specifications, and design. This category of items serves as a plan for the product in the initial stages of a project and as documentation in later stages. Note that description data should be flexible enough to include software analysis and design information, user documents, test plans, and anything else needed. Implementation data consists of the deliverable components of a product. This includes code and documentation for the end user. Verification data consists of correctness information (typically testing information). Maintenance data consists of information used in ensuring continuing usefulness of the project after initial delivery.

Process data is subdivided into **management**, **coordination**, and **quality control** categories. Management data is used to control the project in terms of time and resources used. Coordination data is used to help personnel communicate, thus increasing quality and productivity. Quality control data is used to ensure and generally support development of a correct, robust, safe product.

Another term which appears in this report is *component*. This is a general term referring to an element of unspecified type or a group of elements. A component usually refers to a part of the deliverable product (e.g., a code module or a document). A component may also be part of a specification, design, etc. which refers to code or documents.

PRODUCT

Product Description - Planning, development, documentation of all aspects of the specific product. This is the major category that includes most of what we think of when we think of what the software does.

Functionality - What the product must do. This information should reflect the requirements and specifications for the software. It can be in a formal, semi-formal, or just a natural language format. It should include data input, data output, product behavior, and other properties such as portability and security.

Interfaces - Interaction with external systems. This information should detail what external systems are related to this software and the specific types of interactions between the software and the external systems.

Performance - Time and space that the product uses. This is information that describes the required memory and disk space for the software, along with standard (or typical) execution times. The information may be quite complicated if the software can be run in various size configurations or if execution times are varied dependent on input parameters.

Time Constraints - Real time limitations. This information outlines the time performance constraints placed on the software. This includes any partial or total constraints placed on execution times.

Fault Tolerances - Error and failure handling. This information outlines the acceptable responses of the software to "erroneous" input or to hardware failure. Such errors and failures can include exceptions, faults, and resource limitations. The information in this category can include the types of error messages that are to appear, the kinds of errors that need not be detected, and the kinds of recoveries expected from certain errors.

Data Flow - Movement of data in and out of components or stores. This information describes the way in which data moves throughout the software. It treats each component or store as a data-handling entity and describes that data that moves in and out of that entity (including what the data is, where it came from, and where it is going).

Process Flow - Execution progression of components; sequential/parallel. This information describes the software from a control flow viewpoint discussing the flow of execution in both normal and abnormal situations. It also includes sequential and parallel control flow information.

Resources - Resource usage of component; hardware considerations. Resources are any entities external to the software. This information discusses resources that either supply information to the software or receive information from the

software.

Structure - Static decomposition of components. This information conveys any logical grouping of components for any reason; for example, grouping all components that deal with the same database. There may be several static decompositions for the same software.

Entity-Relationships - Relationships among components and externals. This information includes all of the typical E-R type information, e.g., for each entity to what other entities is it related and in what manner.

Communication - Internal interfaces. Within the software how is communication accomplished? What messages (in the object-oriented sense) are communicated among the software's entities?

Data - Often now being called the **Data Dictionary**, **Data Encyclopedia**, or **Data Repository**. Data types, operations, constants, descriptions, stores, relationships, objects and classes, processes, data flows, events, states, external entities. May be related to Project Index data (see Process Coordination).

Requirements/Design - Relationships of goals and components. This information tells which requirements are related to (satisfied by) which elements of the design.

Design/Performance - Relationships of structure and performance. This information tells which elements of the design are related to the various performance constraints.

Description/Implementation - Relationships of planned and implemented components. This information links the requirements and specifications (description of the software) with the actual implementation. That is, what components implement the requirements and specifications.

Design/Design - Relationships of alternative design representations. For software with more than one design proposed, how does each relate to the other? What are the functionality and performance tradeoffs of each?

Prototypes - What prototyping activity is planned? What specific aspects of the software is to be prototyped? What will be done with the prototype? What simulations will be conducted? What experiments will be tried to test requirements, specifications, design, etc. This information, when complete, should include the prototype goals (questions the prototype is designed to answer) and results (experimentally-determined answers), as well as the actual prototype product, simulation code, etc.

Product Implementation - This is the major category that includes the actual software product (i.e., code, documents, etc.) as well as relevant information.

Actual Product - Code, Documents for end user. This is the software and documentation produced. It consists of all new (and possibly re-used) code and the text and graphics necessary to produce documentation for the software. This category is closely related to Configuration (see Process Coordination) which keeps track of versions, revisions, etc.

Metrics - Product statistics. This information consists of any and all metrics computed primarily from the software code (but possibly also from documentation or other related representations of the product). It may include (but is not limited to) such metrics as lines-of-code, size of data structure, and complexity (e.g., v(G)). Such metrics may be used for management, testing, maintenance, performance, and even quality control purposes.

Library - Globally available, re-usable components. This information contains either actual re-usable components (or some sort of pointer to them) that will be (or have been) employed in the implementation of the software. Such a library may have project, company, or even wider scope.

Templates - Outlines and examples of common components. This information contains sample components that conform to project, company, or wider standards. Such components may simply be bare-bones schema with little actual code or may be nearly complete components that require only minor modification before use in the software.

Compile Parameters - How code is compiled for testing, debugging, and (ultimately) for generating a production version. This information includes standard compilation parameters, ways of testing various versions, searching order for external components (such as re-used components), and special parameters necessary for preparing the product version.

Product Verification - This is the major category that includes all information related to testing the software (or any related activity that attempts to discover and correct errors).

Test Plan - Outline of testing process. This contains at least rudimentary information about how the software is to be tested: what types of testing procedures (perhaps formal methods) will be pursued, what tools will be used, what types of test data, what will be done about errors that are discovered, etc.

Test Tools - Custom functions for debugging and testing. This is information about the specific tools that will be (were) used for testing the software. These can include tools that are part of the CASE tool, standalone external tools, or specific test harnesses to be produced as part of the software development process.

Test Suites - Test data and expected results. This information describes specifically how test data is to be generated, how the software is to be "exercised" with this data, and how the results are to be interpreted.

Status - This information (collected during the software testing process) outlines which tests have detected the presence of an error and which tests have failed to detect the presence of an error. Obviously, it is possible to tell from this information which tests have been run (and either detected or failed to detect errors) and which tests have not been run. For regression tests, this information will tell which have been run on which versions and which revisions.

Errors Found - Errors discovered through testing; error reports. This information outlines what errors have been discovered, which have been corrected, which are planned to be corrected, and which (if any) are not planned to be corrected.

Verification/Description - This information links the requirements and specifications (description of the software) with the verification process. That is, what has been (will be) done to assure that specific requirements and specifications have been tested.

Analysis - Results of matching implementation against description (i.e., requirements and specifications). This information includes such items as types of errors, time and space performance, error and failure handling, consistency, and completeness.

Product Maintenance - This is the major category that includes all information related to the maintenance of the software, its upkeep, and support of the product in use (and perhaps even in late development stages).

Maintenance History - This information includes all actual changes made and known problems not yet corrected. It also includes information about various software releases and versions and how they differ from each other.

Special Cases - How the product is being used. How the product is being customized. This includes any release or version related information not included in the Maintenance History sub-category above due to special circumstances.

Complaints - Reported errors and their locations, problems; evaluations; replies. This information includes all requests for changes to the software based on actual errors (i.e., the software fails to meet one of its requirements).

Proposed Changes - Reported desires for new versions (including specific modifications); evaluations; replies; planned upgrades. This information includes all requests for changes to the software based on enhancements (i.e., the software meets its requirements, but it could do something even more useful for the end user).

General Information - Any other information related to the software as it is in operation; for example, (but not limited to) market penetration, customer addresses and contacts, and versions and licenses.

PROCESS

Process Management - Resource management for the software project. This is the major category that includes most of the management information pertaining to the software development process. A good CASE tool should support most of the information maintained and manipulated by good stand-alone project management tools.

Schedule - Time to finish each task. This information will include both estimates of task durations and triggering mechanisms (for those not yet completed) as well as actual start, stop, and duration times (for those tasks already completed). It will include any relevant dependency and status information, as well.

Budget - This information includes estimates of salaries, personnel costs, hardware costs, etc. (for tasks not yet completed) as well as actual salaries, personnel costs, and hardware costs (for those tasks already completed). It will include any relevant dependency and status information, as well.

Personnel Assignments - This information includes responsibilities (who is responsible for each aspect of the software development), backups (who are available to step in for those with primary responsibilities), authorities (who has read/write access to what project data), as well as individual data (experience, skills, etc.) for each member of the software development team.

Environment Customization - This information describes the environment in which this project is being developed (including how it may differ from the standard software development environment in this company). What procedures, tools, techniques, languages, management standards, coding standards, and documentation standards are being used. How text and graphics are formatted for various media. This information outlines how the software is to (does) interact with the end users. Information such as standard screen formats, standard error formats, standard input "forms" are all included in this information.

Format Parameters - Parameters for input to and output from the CASE system, including reports throughout the software life cycle that keep management informed of the progress on this software project. What reports are to be generated, what schedule is to be followed for them, are they to be manually or automatically generated, how should they look for various media.

Process Plan - What plan is to be (was) followed in developing the software. What phases are to be employed, what standards, and overall schedule. This can even include pre-project bidding and contracting information and some allowance for process improvement.

Process Coordination - This major category includes all information needed by the software development team for cooperation, communication, and organization.

Project Directory - Project, company, or environment scope directories. This information includes all linkages to people, requirements, specifications, design, code, and testing relevant to this software project. For example, in the people category it can include all personnel working on the project, personnel with previous experience on this or a similar project, personnel with consulting capabilities outside the project, etc.

Configuration - Arrangement of all product and some process data. This includes such information as (but is not limited to) software versions, revisions (history of the software), structural relationships, and control locks (overwrite protection).

Standards - Project consistency rules. This information includes all standards that are to be (were) followed during software development. Note that several other categories include some standards. In this category they are to be all collected including documentation (perhaps the most important), personnel, design, coding, messaging, and implementation standards.

Communication - Intra-group communication. This information includes names, addresses, phone numbers, e-mail addresses, and office locations of all personnel working on the project. It can also include (but is not limited to) mail aliases (mailing lists), note logs, meeting minutes, note/component relationships (i.e., topical index for notes, references).

Communication Formats - Idea communication media. This includes information on the various modes of communication among software development team members: for example, (both in-person as well as electronic versions of the following) forums, bulletin boards, brainstorming sessions, votes, etc.

Process Quality Control - This major category includes all information pertaining to quality assurance including product quality, process quality, run-time environments, and history.

Quality Goals - Criteria to measure quality. This includes information from requirements, specifications, and otherwise that can be used to assess the quality of the completed software project.

Fault Consequences - *What happens if the product fails. This information* describes the severity of the problems involved if the entire product or any components thereof fail to operate according to expectations.

Target Environment - How will the product be used. The software must operate within certain hardware and software constraints. This includes such information as the type of operating system, LAN operation, possible abuses, etc.

Inspections - Standards, schedules, participants, results. This includes information about what inspections are planned (or for a completed project, what inspections were conducted). It also includes information on classes, design meetings, problem resolution meetings, and informal meetings.

User Input - Customer/End-user evaluations and comments. What user input is going to be (was) collected. How is it to be used. What effect will it have on the developing and completed software product. What input will it obtain from experts in the field.

References - Miscellaneous, external references. This can include (but is not limited to) references to similar projects, projects in the same application area, projects conducted for similar hardware systems, projects developed by the same or similar software development teams, etc.

Project History - Record of changes and results of the process. This information includes all aspects of project history that is not found in Configuration (see Process Coordination). It may include (but is not limited to) project summaries, post-mortem analyses, process evaluations, and process improvement suggestions.